REMARKS

Claims 1-41 are pending. Claim 29 has been amended. In view of the following remarks, Applicant respectfully solicits allowance of the subject application and furtherance onto issuance.

35 U.S.C. §102(b) and §103(a)

Claims 1-25, 27-39 and 41 are rejected under 35 U.S.C. §102(b) as being anticipated by U.S. Patent No. 5,379,432 to Orton et. al. (hereinafter, "Orton"). Claims 26 and 40 are rejected under 35 U.S.C. §103(a) as being unpatentable over Orton in view of U.S. Pat. No. 5,752,027 to Familiar (hereinafter, "Familiar"). Applicant respectfully traverses the rejections.

Claim 1 is directed to a method of factoring operating system functions that includes defining criteria that governs how functions of an operating system are to be factored into one or more groups. The functions are factored into one or more groups based upon the criteria. Groups of functions are associated with programming objects that have data and methods. The methods correspond to the operating system functions and are effective to provide an object oriented operating system. The programming objects are configured to be instantiated throughout a remote computing system.

The Office asserts InterProcess Communication (IPC) classes of Orton at Col. 26, lines 57-67 and "ReplyAndReceive" at Col. 28, Lines 9-26 for support of the programming object being configured to be instantiated throughout a remote computing system. The Applicant respectfully disagrees. Orton does not disclose, teach or suggest "programming objects being configured to be instantiated throughout a remote computing system" as claimed in Claim 1.

Beginning at page 13 of the subject specification, an exemplary distribution is described of an operating system's resources across one process boundary and one machine boundary in a distributed computing system. In the described example, in relation to FIG. 4, resource object 48 is instantiated in-process (i.e. inside the application's process), resource object 50 is instantiated in another process on the same machine (i.e. local), and resource object 52 is instantiated on another machine (i.e. remote). Use of remote and local resources is described in an additional embodiment beginning at page 22 of the subject specification. An operating system is provided with the ability to determine, based on the specified unique identifier, whether it has the resource that is requested. If it does not, the operating system can ascertain the location of the particular resource and retrieve it so that the application can have the requested resource. The location from which the resource is retrieved can be ***across process and machine boundaries***. As an example, consider the following. If an application asks for a specific version of a "ReadFile" interface, and the operating system does not support that version, the operating system may know where to go in order to download the code to implement the requested functionality. Software code for the specific requested interface may, for example, be located on a web site to which the operating system has access. The operating system can then simply access the web site, download the code, and provide the resource to the application.

Orton is directed to an object-oriented interface for a procedural operating system in the "context of executing the object oriented application 103A on the computer platform 102." *Orton, Col. 8, Lines 21-22*. Orton describes a limited functionality procedural operating system, such as the Mach micro-kernel. *Orton, Col. 5, Line 67 to Col. 6., Line 2*. The executable entity in Mach is known as a

thread. *Orton, Col. 11, Line 25.* Threads use ports to communicate with each other. A port is "basically a message queue inside the kernel if [sic] at threads can add messages to or remove message from, if they have the proper permissions to do so." *Orton, Col. 15, Lines 39-42.* Ports exist solely in the kernel and can only be manipulated via port rights. *Orton, Col 15, Lines 48-49.* Thus, Orton describes that messages are utilized by threads to communicate with each other through the use of ports.

InterProcess Communication (IPC) classes are utilized in Orton to represent the messages for communication between threads, as shown in the following excerpt:

> The IPC classes 410 represent the Mach IPC message abstraction. Note that all messaging behavior is on the message classes; the port right classes are basically for addressing the message. The usage model is preferably as follows: A TIPCMessageStream is instantiated objects are streamed into it, and the TIPCMessageStream::Send method is called with an object representing a destination send-right passed as an argument. To receive a message, a TIPCMessageStream is instantiated and its Receive method called, passing in a receive-right object as an argument. *Orton, Col. 26, Lines 57-56.*

Orton then describes methods for sending/receiving IPC messages, an example of which was asserted by the Office and is excerpted as follows:

> ReplyAndReceive (const TPortSendSideHandle& replyToPort, const TPortReceiveSideHandle& receivePort, MIPCMessage& receiveMsg, const TTime& timeout=kPositiveInfinity) sends back a reply, blocks and receives a new message.
> Subclasses' methods for getting/setting port right fields in header (Remote and Local Ports: On SEND side, REMOTE PORT specifies the destination port, and LOCAL PORT specifies the reply port. On RECEIVE side, REMOTE PORT specifies the reply port (port to be replied to) and LOCAL PORT specifies the port received from. The way the port was

> (or is to be) transmitted is returned in theDisposition. It can have values: MACH.sub.-- MSG.sub.-- TYPE.sub.-- (MOVE.sub.– RECEIVE, MOVE.sub.– SEND, MOVE.sub.-- SEND.sub.-- ONCE, COPY.sub.-- SEND, MAKE.sub.-- SEND, MAKE.sub.-- SEND.sub.-- ONCE}.): GetRemotePort: pass in the remote port right, and specify the disposition. *Orton, Col. 28, Lines 9-26.*

It is respectfully submitted that the Office has misinterpreted the use of "interprocess communication" as indicating remote communication. This is not the case. As shown in the above excerpts, Orton describes communication between threads in the "context of executing the object oriented application 103A on the computer platform 102." *Orton, Col. 8, Lines 21-22.* This is further supported by the Summary of the Invention in Orton, which is excerpted as follows:

> The present invention is directed to a system and method of enabling an object-oriented application to access in an object-oriented manner a procedural operating system having a native procedure interface. *The system includes a computer and a memory component in the computer. Orton, Col. 3, Lines 50-55 (emphasis added).*

Therefore, the system described in Orton is implemented in a single computer. Indeed, nowhere in Orton is remote communication or a machine boundary even discussed. Rather, Orton describes communication between an application and an operating system that are executed on a computer platform, such as "an International Business Machines (IBM) computer or an IBM-compatible computer .... [or an] Apple computer". *Orton, Col. 6, Lines 6-10.* Although Orton describes "interprocess communication", Orton does not disclose, teach or suggest communication in ***a remote computing system*** or ***across machine boundaries***. Therefore, *Orton* does not disclose, teach or suggest "programming objects being

configured to be instantiated throughout a remote computing system" as claimed in Claim 1. According, for at least these reasons, this claim is allowable.

Claims 2-13 depend either directly or indirectly from claim 1 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 1, are neither shown nor suggested in the references of record, either singly or in combination with one another. For example, claim 7 recites "instantiating a plurality of programming objects across a machine boundary". Orton does not disclose, teach or suggest the presence of a machine boundary nor even mention more than one computer platform. Therefore, for at least this reason, claim 7 is allowable.

Claim 14 recites a method of factoring operating system functions. A plurality of operating system functions that are used in connection with operating system resources into first groups based upon first criteria. The first groups are also factored into individual sub-groups based upon second criteria. Each sub-group is assigned to its own programming object interface. A programming object interface represents a particular object's implementation of its collective methods that are effective to provide an object-oriented operating system. Individual objects having associated programming object interfaces are configured to be instantiated throughout a remote computing system.

As previously described, Orton neither discloses nor suggests a method that factors operating system functions where "individual objects having associated programming object interfaces are configured to be instantiated throughout a remote computing system" as claimed in claim 14. Accordingly, for at least this reason, claim 14 is allowable.

Claims 15-23 depend either directly or indirectly from claim 14 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 14, are neither shown nor suggested in the references of record, either singly or in combination with one another.

Claim 24 recites a method of factoring operating system functions which includes factoring a plurality of operating system functions into interface groups based upon the resources with which a function is associated. The interface groups are factored into interface sub-groups based upon each function's use of a handle that represents a resource. The interface sub-groups are organized so that at least one of the interface sub-groups inherits from at least one other of the interface sub-groups. Individual interface sub-groups are associated with individual programming objects that can be instantiated throughout a remote computing system.

Again, as previously described, Orton neither discloses nor suggests a method that factors operating system functions where "individual objects sub-groups being associated with individual programming objects that can be instantiated throughout a remote computing system" as claimed in claim 24. Accordingly, for at least this reason, claim 24 is allowable.

Claims 25 and 27-28 depend from claim 24 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 24, are neither shown nor suggested in the references of record, either singly or in combination with one another.

Claim 26 is rejected over Orton in view of Familiar. Familiar does not cure the defects of Orton, namely "individual objects sub-groups being associated with individual programming objects that can be instantiated throughout a remote computing system" as claimed in claim 24. Therefore, given the allowability of claim 24, claim 26 is allowable over both Orton and Familiar, alone and in combination.

Claim 29 has been amended and, as amended, recites an operating system application program interface embodied on a computer-readable medium comprising a plurality of object interfaces. Each object interface is recited to be associated with an object that includes one or more methods that are associated with and can call functions of an operating system that does not comprise the object interfaces. At least one object is configured ***to be remotely instantiated***. Nowhere does Orton disclose or even suggest any such subject matter. Accordingly, this claim is allowable.

Claims 30-35 depend either directly or indirectly from claim 29 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 29, are neither shown nor suggested in the references of record, either singly or in combination with one another.

Claim 36 recites an operating system that includes a plurality of programming objects having interfaces. The programming objects represent operating system resources, and the interfaces define methods that are organized in accordance with whether they create an operating system resource or not. The programming objects are configured to be called either directly or indirectly by an application. The methods are configured to call operating system functions

responsive to being called directly or indirectly by an application. The programming objects are configured to be instantiated throughout a remote computing system.

As previously described, Orton neither discloses nor suggests "programming objects being configured to be instantiated throughout a remote computing system" as claimed in claim 36. Accordingly, for at least this reason, claim 36 is allowable.

Claims 37-39 depend from claim 36 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 36, are neither shown nor suggested in the references of record, either singly or in combination with one another.

Claim 40 is rejected over Orton in view of Familiar. Familiar does not cure the defects of Orton, namely "programming objects being configured to be instantiated throughout a remote computing system" as claimed in claim 36. Therefore, given the allowability of claim 36, claim 40 is allowable over both Orton and Familiar, alone and in combination.

Claim 41 recites a method of converting an operating system from a non-object-oriented format to an object oriented formal. The operating system includes a plurality of operating system functions that are callable to create or use operating system resources. A plurality of programming object interfaces are defined that define methods that correspond to the operating system functions. Programming objects that support the interfaces are callable either directly by an application that makes object-oriented calls, or indirectly by an application that makes function calls. The programming objects being configured to be

instantiated throughout a remote computing system. A programming object interface is called either directly via an object-oriented call, or indirectly via an indirection that transforms a function call into an object-oriented call. Responsive to the calling, an operating system function is called with a method of the programming object that supports said programming object interface. Orton neither discloses nor suggests "programming objects being configured to be instantiated throughout a remote computing system" as claimed in claim 41. Accordingly, for at least this reason, claim 36 is allowable.

For at least these reasons, claims 1-25, 27-39 and 41 are allowable over Orton. Applicant respectfully requests that the §102 rejection of claims 1-25, 27-39 and 41 be withdrawn.

Additionally, Claims 26 and 40 are allowable over Orton in view of Familiar. Applicant respectfully requests that the §103 rejection of claims 26 and 40 be withdrawn.

Conclusion

All pending claims 1-41 are in condition for allowance. Applicant respectfully requests reconsideration and prompt issuance of the subject application. If any issues remain that prevent issuance of this application, the Examiner is urged to contact the undersigned attorney before issuing a subsequent Action.

Respectfully Submitted,

Dated: December 11, 2003 By: _____

William J. Breen, III
Reg. No. 45,313
(509) 324-9256 x249